

- RAM modulari
  - usano dispositivo a 3 stati

**3-state**

i	e	u
0	0	non connesso
0	1	0
1	0	non connesso
1	1	1

Lezione 5      Architettura degli Elaboratori - 1 - A. Sperduti      Pagina 30

## Microarchitettura: Livello 1

- Sistema visto come composizione di **n unità di elaborazione interagenti ed autonome** (dati gli ingressi ottenuti tramite interazione con altre unità)
- Ad ogni unità è affidato un certo **sottinsieme** delle funzionalità del sistema
- L'interazione tra tali unità fornisce la funzione complessiva di sistema
- Un elaboratore generico comprende molte unità (CPU, memoria, I/O, etc.)

Lezione 5      Architettura degli Elaboratori - 1 - A. Sperduti      Pagina 32

CS = 0 chip non attivo    CS = 1 chip attivo    R/nW=1 lettura    R/nW=0 scrittura

Lezione 5      Architettura degli Elaboratori - 1 - A. Sperduti      Pagina 31

## Livello 1 (segue)

- Il livello 1 è anche detto **livello firmware**
- Il livello firmware è interpretato direttamente (eseguito) dall'hardware (livello 0: circuiti digitali)
  - mediante l'interconnessione di reti combinatorie e sequenziali
- Ciascuna unità opera in modo **Sequenziale**, con funzionamento specificato da un *microprogramma* ( $\mu$ programma) espresso in un dato *micro-linguaggio* ( $\mu$ linguaggio) che permette di programmare la interazione fra le reti combinatorie e sequenziali del livello 0

Lezione 5      Architettura degli Elaboratori - 1 - A. Sperduti      Pagina 33

## Livello Firmware

- Ogni unità a livello firmware è in grado di eseguire sequenze di operazioni
- Le operazioni che formano tali sequenze, dette operazioni esterne dell'unità, sono **indipendenti fra loro** ed appartengono ad un insieme predefinito
- L'esecuzione di ogni diversa operazione esterna è descritta da un diverso frammento di *μprogramma* (detto interprete dell'operazione)
- Esiste anche una operazione speciale che di fatto realizza l'acquisizione dall'esterno dei dati e del codice della operazione esterna da eseguire

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 34

## Livello Firmware

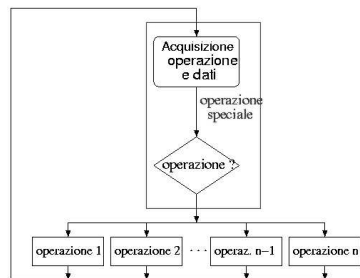
- L'interpretazione del microprogramma viene usualmente eseguita da due reti sequenziali LLC interagenti, denominate **Parte Controllo (PC)** e **Parte Operativa (PO)**
- PC è l'automa di controllo dell'unità
  - ha tanti stati interni quante sono le μistruzioni del μprogramma (lo stato corrente determina la μistruzione in esecuzione)
- PO esegue le operazioni elementari previste da ogni μistruzione
  - quindi dispone di: registri, reti combinatorie (ALU, ...), circuiti per l'instradamento dei dati (multiplexer, bus, ...)

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 36

## Livello Firmware



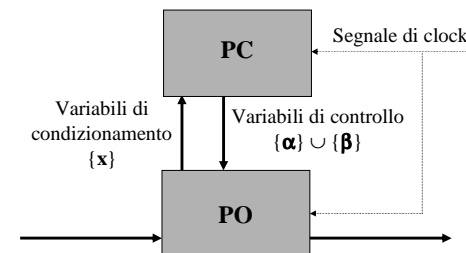
- Il *μprogramma* di una unità riunisce i frammenti di programma delle diverse operazioni (esterne e speciale)
- Il *μprogramma* ha una struttura ciclica in cui si alterna l'esecuzione della operazione speciale con l'esecuzione della operazione esterna il cui codice e dati da elaborare sono stati acquisiti dalla operazione speciale

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 35

## Livello Firmware (segue)



Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 37

## Livello Firmware (segue)

- La PO provvede all'esecuzione dei comandi del microlinguaggio (**microistruzioni**) tramite reti combinatorie standard e registri
- La PC provvede a:
  - Controllo di sequenzializzazione delle microistruzioni tramite **variabili di condizionamento**  $\{x\}$  relative allo stato interno di PO
  - Invio comandi di esecuzione a PO tramite **variabili di controllo**  $\{\alpha\} \cup \{\beta\}$

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 38

## Livello Firmware (segue)

- **Progettazione di unità**
  1. Specifica delle operazioni esterne dell'unità
  2. Scrittura del microprogramma di interpretazione delle operazioni esterne
  3. Derivazione di PO a partire dal microprogramma
  4. Derivazione di PC a partire dal microprogramma
  5. Determinazione del periodo di clock
  6. Valutazione delle prestazioni

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 40

## Livello Firmware (segue)

- Variabili di condizionamento  $\{x\}$  : PO  $\rightarrow$  PC
  - Esempio: test per zero del contenuto di un certo registro
- Variabili di controllo : PC  $\rightarrow$  PO
  - $\{\beta\}$  abilita / disabilita la scrittura nei registri di PO
  - $\{\alpha\}$  fornisce gli ingressi secondari per i commutatori, selezionatori, ALU, etc. di PO
    - Esempio: instradamento da registro/i sorgente a registro destinazione della microistruzione

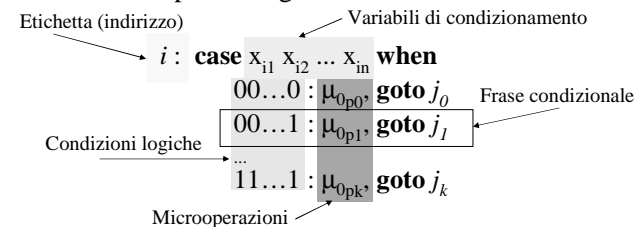
Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 39

## Livello Firmware (segue)

- Ogni microistruzione del microprogramma esegue in un ciclo di clock
- Microlinguaggio Phrase Structured (**PS**)
- Microoperazioni nulle (*nop*) o di trasferimenti, anche multipli, tra registri



Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 41

### Livello Firmware (segue)

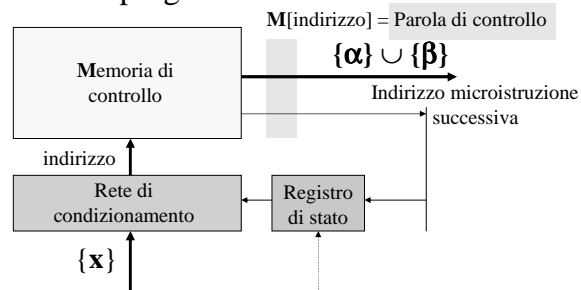
- **PO** tipicamente vista come rete sequenziale di **Moore**
  - Per microoperazioni non parallele basta una ALU multi-funzione
- **PC** come rete sequenziale di **Mealy**
  - Etichette microistruzioni  $\Leftrightarrow$  stati interni
  - Variabili di condizionamento  $\Leftrightarrow$  stati ingresso
  - Microoperazione (=variabili di controllo)  $\Leftrightarrow$  stati di uscita

### Livello Firmware (segue)

- Controllo residuo
  - In aggiunta alle variabili di controllo  $\{\alpha\} \cup \{\beta\}$  emesse dalla PC
  - Generate come funzione del contenuto dei registri della PO
  - Riducono la complessità di struttura e di progettazione della PC

### Livello Firmware (segue)

- PC microprogrammata



- $\{\alpha\} \cup \{\beta\}$  designa la microoperazione richiesta

### Modelli PC-PO alternativi

- **Moore-Moore**, microlinguaggio Transfer-Structured (TS)
  - Maggior numero di cicli di clock ma di minor durata rispetto al modello Mealy-Moore
- **Moore-Mealy**
  - Equivalente al modello Moore-Moore, con PO ottenuta *anticipando* il prelievo di  $\{x\}$  all'ingresso dei registri

## Livello Firmware (segue)

### Microlinguaggio TS

```

etichetta : microoperazione
case condizione logica
  when valore 1 => indirizzo successivo 1
  when valore 2 => indirizzo successivo 2
  ...
  when valore n => indirizzo successivo n
end case
    
```

## Esempio di PC-PO

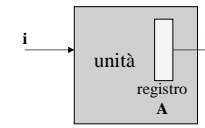
### Assunzioni:

- una sola operazione esterna (vedi specifica sotto) che richiede un ingresso **i** (numero rappresentato in complemento a 2)
- per semplificare non si considera l'operazione speciale, cioè si assume che:
  - l'unità riceva l'ingresso **i** quando l'unità è pronta per l'elaborazione
  - l'uscita **u** della unità (memorizzata internamente in un registro **A**) è prelevata da chi ne ha bisogno prima che questa venga sovrascritta dalla uscita successiva

### Specifica:

```

while true loop
  A := A + i; /* i ingresso */
  if A < 0 then A := -A end if;
end loop;
    
```



### PS (Mealy-Moore): [non ottimizzato]

```

0. A + i → A, 1
1. (A0 = 1) -A → A, 0 ; (A0 = 0) nop, 0
    
```

## Esempio

### Specifica:

```

while true loop
  A := A + B;
  if A < 0 then A := -A end if;
end loop;
    
```

### Formalismi di µprogramma:

; : azioni *alternative*  
 , : azioni *parallele*  
 Le condizioni logiche si *omettono* se *tutte* abilitano la *stessa* azione

### PS (Mealy-Moore): [ottimizzato]

```

0. A + B → A, 1
1. (A0 = 1) -A → A, 0 ; (A0 = 0) A + B → A, 1
    
```

### TS (Moore-Moore):

```

0. A + B → A, 1
1. nop (A0 = 0) 0 ; (A0 = 1) 2
2. -A → A, 0
    
```

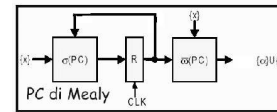
### TS (Moore-Mealy):

```

0. A + B → A (A0 = 0) 0 ; (A0 = 1) 1
1. -A → A, 0
    
```

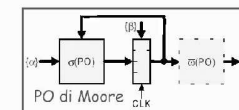
{x} anticipato

## PC-PO: Mealy-Moore

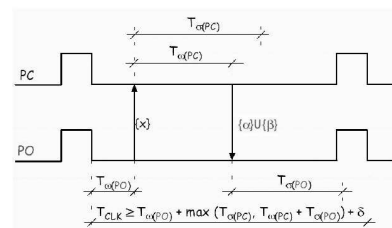


I segnali di condizione derivano dalle uscite dei registri della PO

$\omega(PO)$  può mancare o essere molto semplice



• Funzionamento seriale



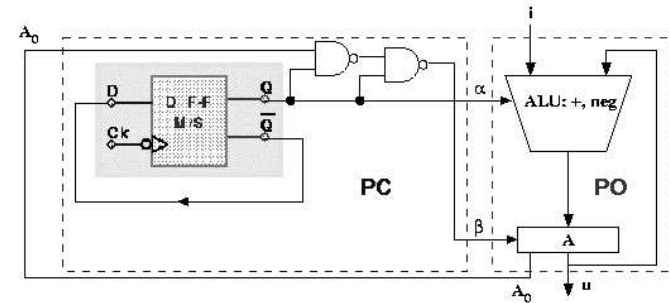
- $T_{CLK}$  è il periodo di clock
- $T_{\alpha(PC)}$ : ritardo massimo della rete combinatoria  $\sigma(PC)$
- $T_{\alpha(PO)}$ : ritardo massimo della rete combinatoria  $\sigma(PO)$
- $T_{\alpha(PC)}$ : ritardo massimo della rete combinatoria  $\omega(PC)$
- $T_{\alpha(PO)}$ : ritardo massimo della rete combinatoria  $\omega(PO)$
- $\delta$  ritardo prop.+ setup flip-flop

### Esempio: PO

- La PO ha bisogno di:
  - una ALU in grado di effettuare sia l'addizione ( $\alpha=0$ ) che la negazione (per il cambio di segno;  $\alpha=1$ )
    - in ingresso la ALU ha  $i$  (ingresso alla unità) e il contenuto del registro A, ed  $\alpha$  è il segnale di controllo che seleziona l'operazione da eseguire
  - un registro A, la cui scrittura è abilitata con segnale  $\beta=1$  e disabilitata con segnale  $\beta=0$ 
    - il segno del valore contenuto in A è stabilito dal bit  $A_0$  del registro A ( $A_0$  costituisce il segnale di condizionamento  $x$ )
- lo stato interno di PO è costituito dal contenuto del registro A (se il registro è a  $n$  bit, allora la PO avrà  $2^n$  stati interni)
- la funzione di uscita di PO (modello Moore) è l'identita (infatti l'uscita  $u$  coincide con il contenuto del registro A, cioè lo stato)

### Realizzazione di PC-PO

- Solo 2 stati: uno per ogni istruzione  $\rightarrow$  si usa un flip-flop D
- Usando la tabella di PC si ottiene il seguente circuito



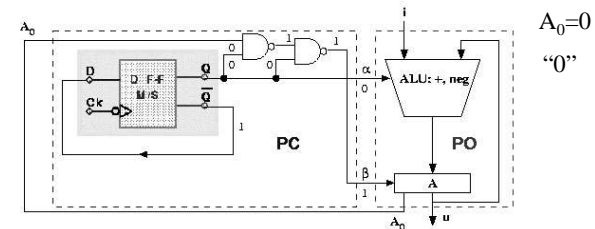
### Esempio: PC

- La PC è una rete sequenziale di Mealy dotata di 1 ingresso binario ( $A_0$ ), 2 uscite binarie ( $\alpha$ ,  $\beta$ ) e 2 stati interni ("0", "1")
  - nello stato "0", qualunque sia il valore assunto da  $A_0$ , la PC deve generare i segnali  $\alpha=0$  ( $A+i \rightarrow A$ ) e  $\beta=1$  (registro A abilitato)
  - nello stato "1", se  $A_0=1$ , la PC deve generare i segnali  $\alpha=1$  ( $-A \rightarrow A$ ) e  $\beta=1$  (registro A abilitato)
  - nello stato "1", se  $A_0=0$ , la PC deve generare i segnali  $\alpha=*$  (non importa il valore di  $\alpha$ ) e  $\beta=0$  (registro A disabilitato)

Stato \ Ingresso	$A_0=0$	$A_0=1$
"0"	"1" $\alpha\beta=01$	"1" $\alpha\beta=01$
"1"	"0" $\alpha\beta=*0$	"0" $\alpha\beta=11$

Tabella della PC

### Esempio PC-PO



Stato \ Ingresso	$A_0=0$	$A_0=1$
"0"	"1" $\alpha\beta=01$	"1" $\alpha\beta=01$
"1"	"0" $\alpha\beta=*0$	"0" $\alpha\beta=11$

Tabella della PC

### Esempio PC-PO

$A_0=1$   
"0"

Stato \ Ingresso	$A_0=0$	$A_0=1$
"0"	"1" $\alpha\beta=01$	"1" $\alpha\beta=01$
"1"	"0" $\alpha\beta=*0$	"0" $\alpha\beta=11$

Tabella della PC

Lezione 5      Architettura degli Elaboratori - 1 - A. Sperduti      Pagina 54

### Esempio PC-PO

$A_0=1$   
"1"

Stato \ Ingresso	$A_0=0$	$A_0=1$
"0"	"1" $\alpha\beta=01$	"1" $\alpha\beta=01$
"1"	"0" $\alpha\beta=*0$	"0" $\alpha\beta=11$

Tabella della PC

Lezione 5      Architettura degli Elaboratori - 1 - A. Sperduti      Pagina 56

### Esempio PC-PO

$A_0=0$   
"1"

Stato \ Ingresso	$A_0=0$	$A_0=1$
"0"	"1" $\alpha\beta=01$	"1" $\alpha\beta=01$
"1"	"0" $\alpha\beta=*0$	"0" $\alpha\beta=11$

Tabella della PC

Lezione 5      Architettura degli Elaboratori - 1 - A. Sperduti      Pagina 55

### Alternativa

- PC microprogrammata

- $\{\alpha\} \cup \{\beta\}$  designa la microoperazione richiesta

Lezione 5      Architettura degli Elaboratori - 1 - A. Sperduti      Pagina 57